A self organized holonic control for mechatronics complex systems: application to a robotized car park

Patrick Pujo, Fouzia Ounnar, Cecilia Zanni

Université Paul Cézanne
Laboratoire des Sciences de l'Information et des Systèmes - UMR CNRS 6168
Avenue Escadrille Normandie Niemen, 13397 Marseille cedex 20, France
{patrick.pujo, fouzia.ounnar, cecilia.zanni}@lsis.org

Received May 28, 2005

Abstract: *In this paper, we describe the two conditions so that a complex system, composed of several mechatronics machines, may be qualified as a mechatronics system. The first condition reflects the aptitude of these machines to work together without the intervention of a central decision system of higher hierarchical level. The second reflects their aptitude to manage their own behavior and to generate the tasks to be carried out in the context of their execution. A self organized holonic control accomplishes these conditions. An example of implementation of this approach is presented with the application to a robotized car park.*

1 Introduction

We can define a mechatronics object as being the result of the integration of elementary mechanical and electronic components. The integration means the localized association of these components: the mechatronics product is more compact. It implies that the mechatronics product seems more complex due to the synergetic tangling of those two technologies. In return, it is more effective than a traditional product. In particular, electronics brings to the mechanical engineering all the possibilities of intelligent control. This intelligent control supports numerous and new functions, such as surveillance, diagnosis, communication...

Can a complex system, composed of several mechatronics machines, be itself described as a mechatronics system? Our answer is positive when the component machines are able, in an autonomous and coordinated way, to ensure its operation, without the intervention of a central supervision, command or management system. We speak then about self organized control functions, which are integrated in the intelligent control of each mechatronics machine. We will describe a holarchic approach of such a solution, where the control of each mechatronics machine can be easily equipped with these two particular functions. The first makes it possible to manage the interactions with the other mechatronics machines. The second makes it possible to place the capacities of the machine with respect to what the other machines can do so that the total system can continue to evolve.

After having presented the various conceptual proposals for our approach, we will develop an example of application.

2 Self Organized Holonic Control for Mechatronic Complex Systems

The organization of the decision-making system configures the ability of a complex system for being a mechatronics one.

2.1 Self Organized Holonic Control

The concept of organization must firstly be specified to define the one of self-organization. A production system is organized when each one of its constitutive entities has a completely formalized behavior thus depending only on the stimuli coming from external orders emitted by a supervisor that manages the coordination. This classical approach of decision-making is naturally strongly centralized: the supervisor generates at level N+1 the group of necessary orders to set in motion the entities of level N.

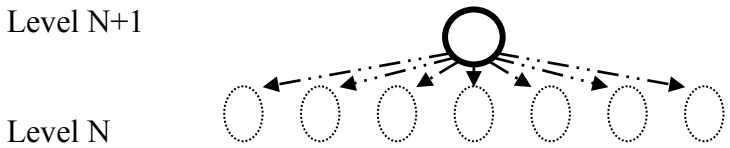

Figure 1: Centralized organization of the decision-making

In the world project IMS (Intelligent Manufacturing System) *(IMS, 1997)*, the concept of self organized system is defined as a "system not-coordinated by outside. The elements are endowed with autonomy and carry out tasks together, in interaction and mutual comprehension; the sum, or combination, of the individual tasks allows to generate an order, a good or a more total service". The concept of self-organization is defined as the process during which structures emerge at the collective level (appearance of a structure on the N+1 scale starting from a dynamics defined on the N scale) starting from a multitude of interactions among individuals.
Three remarks underline the link between autonomy of each decision-making center and its aptitude for self-organization.
Firstly, we must regard the self-organization as a mode of decision-making in real time without preliminary estimated organization. Indeed, if there were a previously organization (with classical control), there would not need more to be self-organized (with intelligent – or mechatronics – control). This implies a temporal horizon of decision-making in a very short term, because the events occurring at every moment can involve brutal ruptures of behavior.
Then, to get an organization, one needs a common goal to these decision-making centers. We can get different versions of this organization, according to the machine properties and the tasks characteristics that we want to organize. For the tasks to be executed, this is translated into terms of synchronization, coordination, cooperation, negotiation and/or generation. A holon is a good concept to represent a decision-making center *(Koestler, A, 1989)*.
The solution finally adopted to make the set of these components operational will be obtained by emergence. Indeed, when there's no hierarchy, each component is involved, on the one hand in the proposal for solutions, and on the other hand in the evaluation of solutions. The most powerful proposal from the point of view of the evaluation criteria is the one to be adopted.
A flat holonic form (Fig 2) largely facilitates its implementation *(Bongaerts et al., 2000)*. Indeed, if the necessary processing to self-organization of many entities can be gathered in a decision-making center, this means that it would be necessary to bring back there all the necessary and relative information to the state of each one of them. Information flow would

be then complex to implement, from the point of view of its quantity and reliability. On the other hand, the localization of processing on the components themselves, nearest to the necessary information, is very interesting, with one communication level.
We define a self organized holonic control in the following way: the decision system of a set of holons $\{h_i\}$ is structured in a flat form where these holons jointly ensure the decisions concerning themselves, without instruction or order coming from the decision-making center of higher level and thanks to functional primitives duplicated on each one of them and in interaction via a common communication protocol.

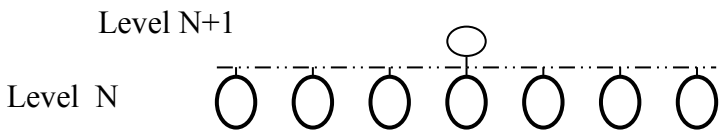

Figure 2: Flat holonic form of the decision-making

This concept of interaction induces the concept of protocol, whose principal function is to specify the organization rules, i.e. the set of potential relations between the system components. Then, that frames a control structure that can be represented into a physical structure and achievable actions by the corresponding equipment.

2.2 Interaction Protocol between Holons

This decision-making mechanism, standard functions of each holon, is based on the competition of the same participant machines according to a Call For Proposals (CFP) launched by an initiator holon. This mechanism is repeated for the execution of each task *(Broissin, N., 1999)*. When a initiator holon H_λ has to execute a new task, the task organization problem consists in determining the holon which will execute it. This holon H_λ broadcasts a Call For Proposals to the others by means of the communication protocol. Each participant holon listens to the message and then uses its task generation function to determine if it can assign this task to itself. In fact, each holon estimates a performance, which depends on common criteria among all participant holons. To estimate this performance, each holon has its own task generation function, which simulates and evaluates the task, taking its load state into account. The holon H_λ compares the result of this simulation with a possible result already released by other participant holons. If its result is better than the released one, then the entity releases its result and then commits itself to execute the future task, while another entity does not release a better result (example: Fig 3).
This task assignment process is inspired by the contract net protocol (CNP) *(Smith, R.G., 1980)*. In classical hierarchical architecture, five basic holons (product, machine, scheduler, computing and negotiation) cooperate as equals *(Kanchanasevee et al., 1997)*. In CNP, the initiator sends out a Call for Proposals. Each participant reviews CFP's and bids on the feasible ones accordingly. The initiator chooses the best bid and awards the contract to the considered participant. Finally, the initiator rejects the other bids. Each initiator thus ensures in a temporarily centralized way management of the CFP which it sent.
Compared to this basic operation, we propose several contributions that come to enrich this protocol and simplify it. Indeed, the idea is to minimize the number of interactions and messages, and to remove any risk of blocking in the case of disturbances in the communication system among holons. The CFP is always launched by the initiator for all the participants, with a deadline. Each participant listens to all the messages concerning this CFP and builds its answer according to the contents of these messages. He answers as soon as he can provide an answer, and only if its proposal is better than those already sent on the

network. If it cannot provide an answer containing a better proposal before the deadline, it never answers. At deadline, all participants and the initiator know the participant who has won the contract and who must perform the task.

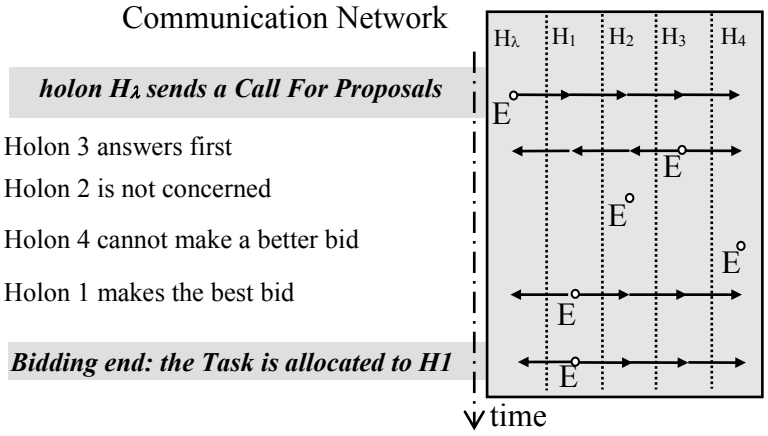

Figure 3: Interaction protocol between holons

The Foundation for Intelligent Physical Agents representation *(FIPA, 2002)* of this new CNP form, called Mechatronics Interaction Protocol (MIP), is shown in Fig 4.

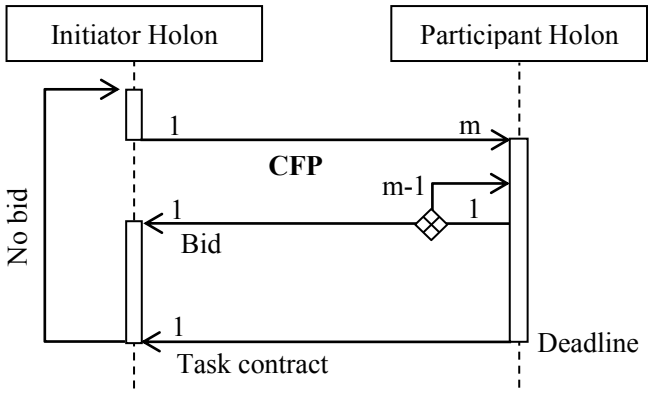

Figure 4: FIPA representation on MIP

If no bid is put forth before the deadline, the CFP must be started again, after, possibly, a modification. Every holon can be an initiator or a participant. Several CFPs are simultaneously in negotiation, since they correspond to tasks to be carried out in the near future on the machines. A holon can be thus simultaneously initiator of a CFP and participant of several others. This interaction protocol gives a strong dynamics to the system control. It confers to the holon an intelligent behavior with respect to the total operation of the system to which it belongs. It is for this reason that we can qualify the holon and its associated machine as ' mechatronics '.

All this decision-making mechanism depends on the capacity of each holon to propose an evaluation of its own performance to carry out a task to come. This is the object of our next paragraph.

2.3 Evaluation of local performance by a task generation function

For each task described in its CFP, the evaluation of each machine performance must hold account of the machine possibilities and the task execution conditions. The task generation function allows the creation of order programs corresponding to the execution of these tasks. These programs can be evaluated. We are thus here intrinsically related to the production function of the machine. For each new type of machine, it is thus necessary to make a fine analysis of its capacities, in order to obtain a model of its own potentially realizable operations. This set constitutes the expertise of the machine in the form of parameterized operations. These operations models are founded on a technological decomposition of the production tasks and their environment, associated with automatic generation algorithms of chronologic actions and elementary trajectories. If the knowledge related to these two aspects may be formalized, then any mechatronics machine with these characteristics can be equipped with a task generation function.
This task generation function is requested twice. The first request is started by the interaction protocol module, in order to give an estimate of the performance with respect to a task being the object of a CFP. The second request takes place once that the task is definitively assigned to the machine, in order to generate the operating sequence which will be actually carried out by it. Two types of task generation functions can be, then, found.
When this function is associated with a NC machine tool, associated knowledge concerns the metal-cutting. For example, in turning, the expertise can be formalized in an algorithmic form. From description of a turning task *(Pujo et al., 1996)*, characteristics of a given NC lathe and the state of its tools, the generation of the tool trajectories by the generation function associated with this lathe will proceed according to following steps: choice of the conditions for catching the piece on the lathe, choice of the tools for each operation, choice of the cutting conditions for each operation, generation of the trajectories of tools and translation into a task program for Numerical Control (ISO language). These algorithms, traditional in CAM, and the associated expertise are described in *(Anselmetti, B., 1990)*. To calculate the obtained performance corresponds to calculate the cost of a transformation by machining. It is necessary to include the cost of the machining itself (which depends on the hourly cost of the machine and the duration of the machining, which itself depends on the continuous paths generated, the power of the machine, the cutting conditions selected…), the cost of the tools (which wear out), the cost of machine setup (which is in fact the cost of immobilization of the resource for a proper configuration), the cost of the raw material, and the cost of latencies during loading and unloading a piece *(Broissin, N., 1999)*. This performance is thus used as a comparison indicator of the various lathes in a robotized turning cell.
When the function of generation of the tasks to be carried out on a machine is function of the current or foreseen tasks on the other machines, it is necessary to take into account the evolution context of the system as a whole. The task generation is carried out according to the present state of each machine and their foreseen tasks in the short term. That is the case when we encounter problems of potential collisions among machines: AGV control problems *(Broissin et al., 1996)* or automated transit rail hub. We will meet this type of generation function in the following example.

3 Application Case: a Robotized Cars Park

The various concepts which we have just presented have to be tested and validated before an implementation on real systems. With that goal, we develop next the simulation model based on the example of a robotized cars park.

3.1 Presentation of a Robotized Cars Park

In the heart of the great urban centers, there are few sites to establish car parks. Moreover, the traditional car parks, where the driver leads his vehicle to the parking place, potentially generate insecurity: robbery, aggression…
An answer can come from the establishment of robotized car parks in the town centers, where the users do not get in and that to equal volume, can store up to twice more vehicles (the places are narrower, the floors are smaller and the circulation spaces occupy fewer surfaces). The architecture of this type of car park is structured around central circulation alleys for mobile robots (Fig 5). On both sides of these alleys there are storage sections, organized in places which can contain one vehicle each. On the same floor, the storage section is composed by two lines of places. Several planes of storage may be superposed, in the air (building) and/or invisible (underground car park). The access to each place is done thanks to a mobile robot, transporting the car according to direction X and circulating between the two lines of places. The passage from one floor to another is done thanks to elevators (direction Z) at each end of the alleys.

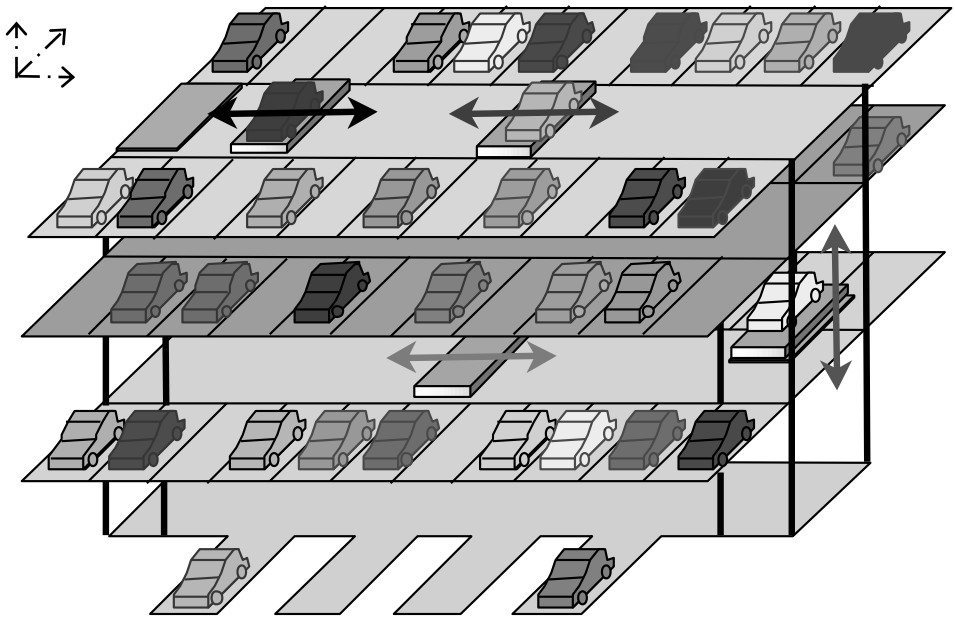

Figure 5: Schematic View of Robotized Cars Park

Input/Output compartments are located at the street level and allow the loading and the unloading of the mobile robots, i.e. the deposit and the recovery of the vehicles by their users. Each mobile robot must provide the following functions:

- to travel in the alleys, to be able to position themselves n front of the places and to get in/out the elevator robots,
- to take or leave a vehicle in a place or an I/O compartment, independently of the dimensions and other characteristics.

A mobile robot, then, consists of two parts. A carrier ensures circulation in the alleys and the access to the elevators (movement according to axis X). This last allows the fine positioning of a unit 'shuttle & vehicle ' opposite the parking places (Fig 6). A shuttle ensures the gripping of the vehicle and can be detached from the carrier to get into in the places and to take or leave the vehicle there (movement according to the axis y).

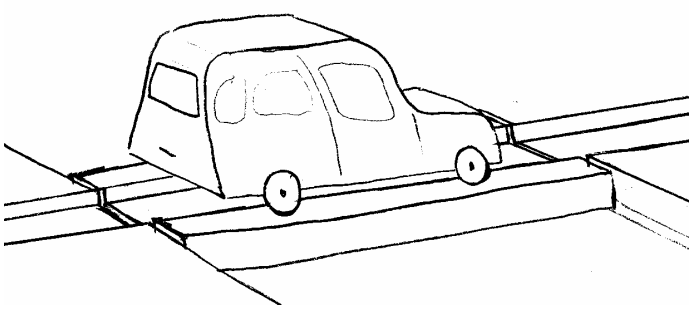

Figure 6: Mobile Robot, facing two places

The shuttle circulates on the carrier and in the places. It passes under the vehicle to seize it by its wheels. It is telescopic to adapt to the distance between the axles of the vehicle.
To arrive to this result, the on-board technology in each mobile robot (MR_i) is highly complex and integrated. The other machines (elevator robots (ER_k) and input/output compartments (IOC_z)) can also be regarded as mechatronics machines. Our contribution consists in equipping them with all the sufficient autonomy so that they do not require a central command system, and that the complete car park may be considered a mechatronics system.
To do so, there will be a communicating computing unit on each machine that will support the holon algorithms: the interaction protocol and the task generation function. It is the latter that will be described next. We will focus on the operation of the mobile robots, which is the most complex.

3.2 Mobile Robot Task Generation

Each MR has to generate its trajectories alone. For that, it has a data structure representing the car park. It knows also the last known position of the other machines, and their estimated movement plan for the current tasks. At time t when it receives the CFP, it has to look for the best possible path. It thus has to be in front of the starting point at the good moment and then it has to reach the arrival point as quickly as possible, to be available again for a new mission. The particular architecture of the car park (an elevator at each end of the alleys) makes the number of possible paths not too high. Unfortunately, these paths are also occupied by other mobile robots, which constitute mobile obstacles to avoid. However, these obstacles are temporary, and it can thus be advisable for the MR to stop waiting for them to disappear. Afterwards, these paths go by the elevator robots, which require again synchronization and waiting. The path taken by the MR must thus be associated to its temporal component to build the MR trajectory. The selected trajectory will be the one with the earliest completion date
For each segment of the path, the initial occupation appears, and therefore the non availability of the segments, as a function of time. This path is analyzed in the plane (x,z), with remarkable positions of the MR, facing the parking places. We will note $p_{k,i}$ this position, with *k* the number of the floor and *i* the index of the corresponding place.
Let us take the example, Fig 7, of the task associated to a car parking. The parking takes place in an input compartment located in position $p_{1,4}$. A free place was reserved in position $p_{5,12}$. The task generation thus consists in finding the best trajectory to go from position $p_{1,4}$ to position $p_{5,12}$, taking into account the above mentioned constraints. We notice that the 5^{th} floor is already occupied by the MR_α, who arrives in the left elevator, it joins position $p_{5,12}$ to leave a vehicle, then moves towards position $p_{5,8}$ to take a vehicle and take it towards the right elevator to, finally, go down again towards the output compartment.

To answer the CFP, the MR_α holon has several solutions to evaluate: going by the left elevator, going by the right elevator or combining both while crossing an intermediate floor. There may also be obstacles on all the possible paths and the task may become impossible for him. We will examine only the first solution.

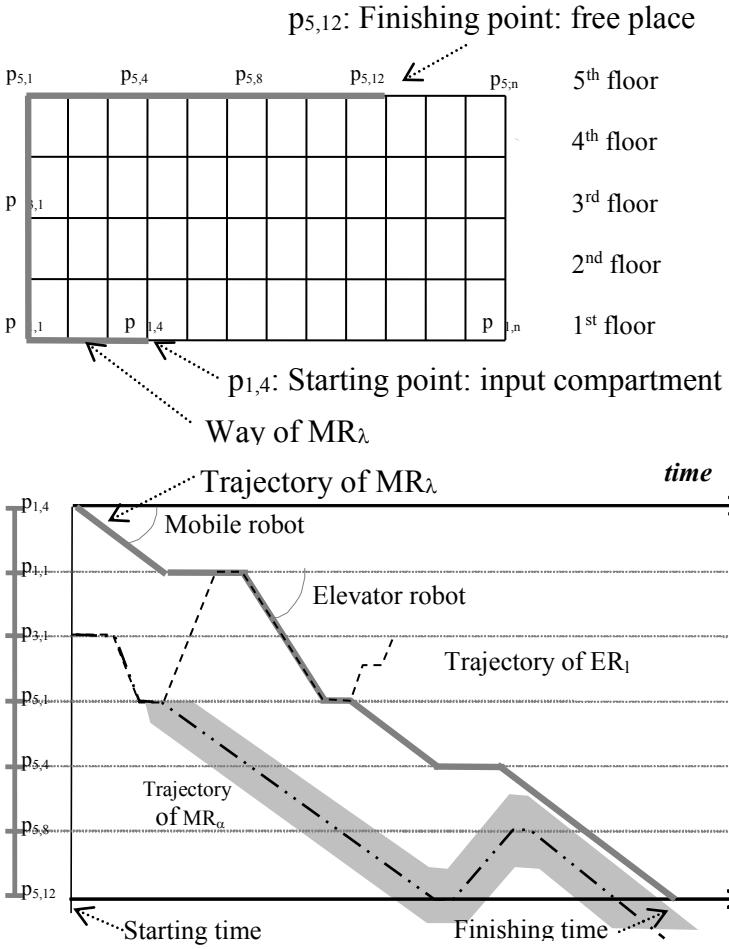

Figure 7: Schematic View of MR trajectories

MR_λ leaves position $p_{1,4}$ and moves towards the left elevator (position $p_{1,1}$). There, it waits for the arrival of the elevator which must be free. For that, it already checked, by a CFP, the availability of the elevators. Once in the elevator, it is brought to the 5th floor, where it gets out (position $p_{5,1}$).

From there, it arrives to the waiting position $p_{5,4}$, so that MR_α completes its operations. As soon as MR_α releases the path, MR_λ follows it by keeping an anti-collision safety distance with it.

Finally, the finishing time of this solution is easy to evaluate. It constitutes the performance of this solution. Each MR will test and evaluate all the solutions relating to it, and will retain the best. It will compare then, via the interaction protocol, this performance with those obtained by the other MR. The most efficient MR will be retained.

3.3 HLA Simulation for MIP Control Validation

For rapidly implementing a flat holonic form, we have used the HLA framework *(IEEE P1516; IEEE P1516.1; IEEE P1516.2)* that is the standard for distributed systems simulation.

The High Level Architecture (HLA) is a software architecture for creating computer simulations out of component simulations. The HLA provides a general framework within which simulation developers can structure and describe their simulation applications.
The HLA was developed by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD) to meet the needs of defense-related projects, but it is now increasingly being used in other application areas. Examples of non-military applications that have already used the HLA are traffic simulations and factory production line simulations.
We can consider a complex simulation as a hierarchy of components of increasing levels of aggregation. At the lowest level is the model of a system component. This may be a mathematical model, a discrete-event queuing model, a rule-based model, etc. The model is implemented in software to produce a simulation. When this simulation is implemented as part of an HLA-compliant simulation, it is referred to as a federate. HLA simulations are made up of a number of HLA federates and are called federations.
The HLA consists of three components: HLA rules, interface specification and Object Model Template (OMT).
At the highest level, the HLA consists of a set of ten HLA rules which must be obeyed if a federate or federation is to be regarded as HLA-compliant. The HLA rules are divided into two groups consisting of five rules for HLA federations and five rules for HLA federates.
The interface specification defines the functional interfaces between federates and the runtime infrastructure (RTI). The RTI is software that conforms to the specification but is not itself part of the specification. It provides the software services, which are necessary to support an HLA-compliant simulation.
The interface specification identifies how federates will interact with the federation and, ultimately, with one another (Fig 8).
Reusability and interoperability require that all objects and interactions managed by a federate and visible outside the federate should be specified in detail with a common format.

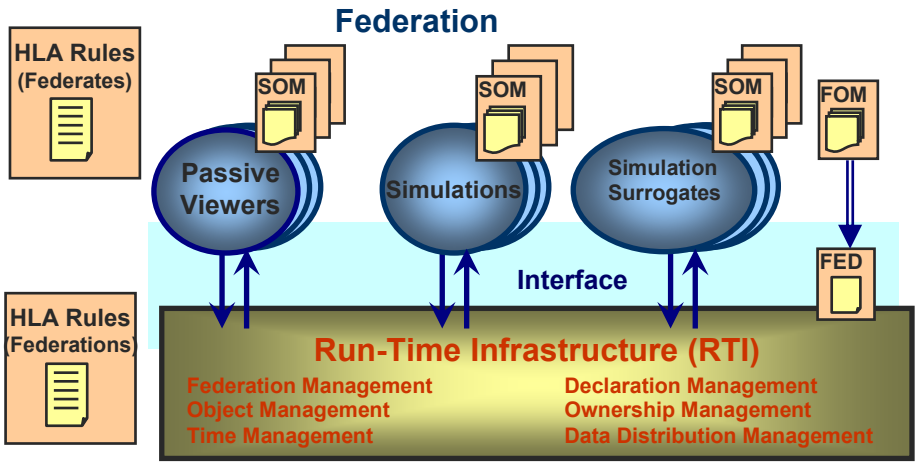

Figure 8: HLA structure

The Object Model Template (OMT) provides a standard for documenting HLA Object Model information. In our simulation environment, every holon is a federate. There are also mobile robot federates, elevator federates and input/output compartment federates. This allows correctly directing the CFP to the correct participants. All these federates communicate with each other via the RTI, over the common interface base provided the interaction protocol.

3.4 Realisation

Practically, a PC network allows, then, to test and validate our flat holonic form. Once the algorithms validated, it is necessary, of course, to implement them on the on-board hardware of the machines. What it important to remark is that either by simulation or on-board, the algorithms will remain the same.
Fig 9 shows an UML Sequence Diagram example, for the following situation: A mobile robot going from the input/output compartment in the same floor to a place in a different floor.

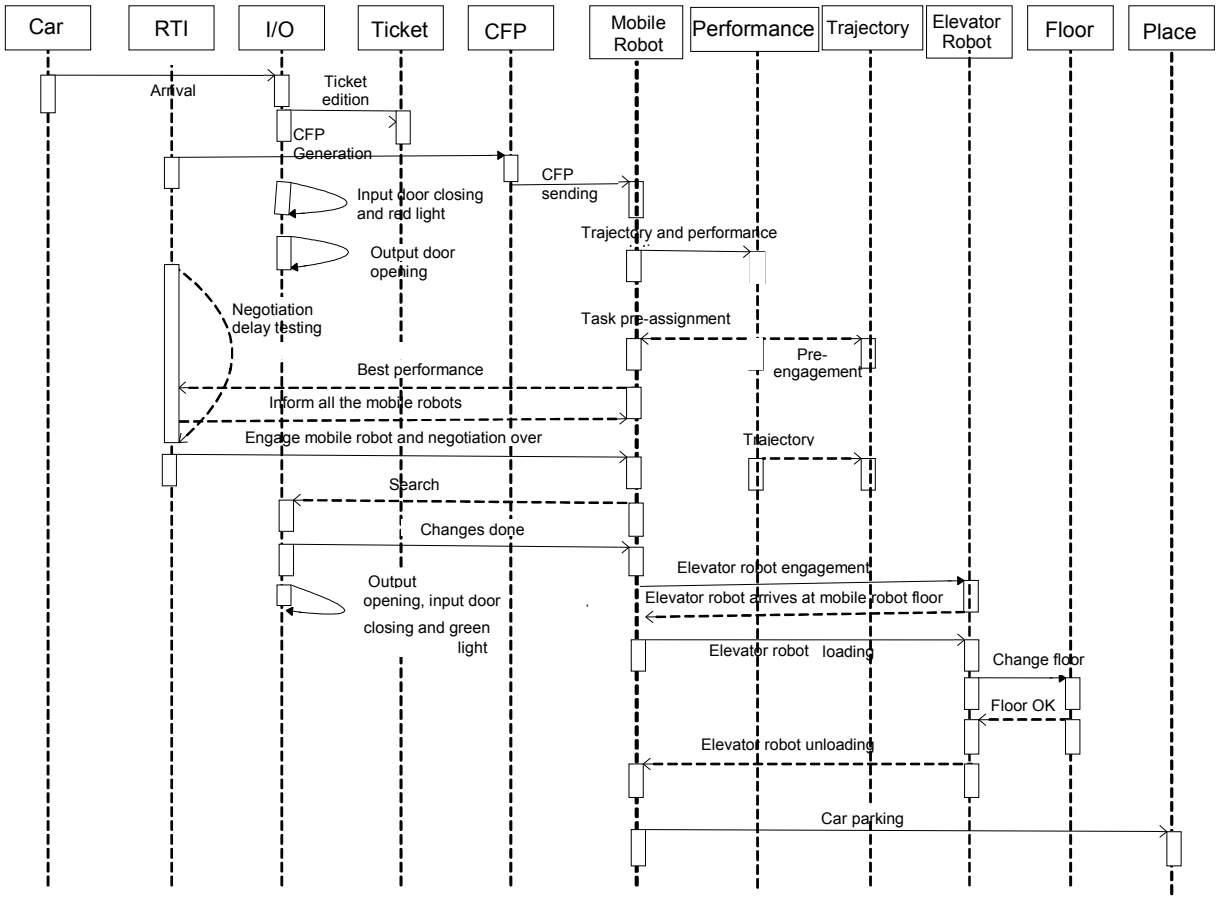

Figure 9: Sequence diagram

The steps presented in this sequence diagram are:

1. The client leaves his car in the I/O compartment
2. The client presses on the desk key at the I/O compartment, for his car being taken care by the automated car park, the desk prints his park ticket.
3. A CFP for car parking is then generated
4. This CFP is sent via the RTI to all the mobile robots in the park
5. The input door of the compartment is then closed and the light turns to red, to indicate that the car is being taken care
6. The output door of the compartment opens to allow the mobile robot load the car to park

As long as the CFP negotiation delay is not finished

7. Every mobile robot calls for its performance module to compute the trajectory and performance for the CFP that it is taking over
8. If the computed performance is better than the one that is booked at that moment, then this mobile robot will be pre-assigned the task and it will also pre-engage the segments of the pre-assigned trajectory

9. The mobile robot that has been pre-assigned the CFP will inform all the others the new performance and the CFP pre-assignment.

CFP negotiation time over

10. Once the CFP negotiation time is over, that mobile robot effectively engages on the task.
11. The mobile robot engages the set of segments that constitute its trajectory
12. The mobile robot moves horizontally up to the I/O compartment
13. The mobile robot loads the car that is in the I/O compartment
14. The output door of the compartment closes, the input door of the compartment opens, the light turns to green

Since the mobile robot is not at the same floor of the parking place, it has to engage an elevator robot

15. The mobile robot asks the elevator robot to come and search it at the floor where he is at that moment
16. The elevator robot indicates its arrival to the mobile robot
17. The mobile robot loads into the elevator robot
18. The elevator robot, loaded with the mobile robot, goes up or down, to the corresponding floor
19. The elevator robot reaches the floor where there is the parking place
20. The elevator robot unloads the mobile robot
21. The mobile robot moves to the parking place to leave the car.

The evolution of the various robots can be graphically observed via Windows. This interface is also useful for manual piloting (vehicle input/output by click on buttons) or automatic (choice of the frequencies of input/output, on average and standard deviation) and makes it possible to visualize indicators of performance.

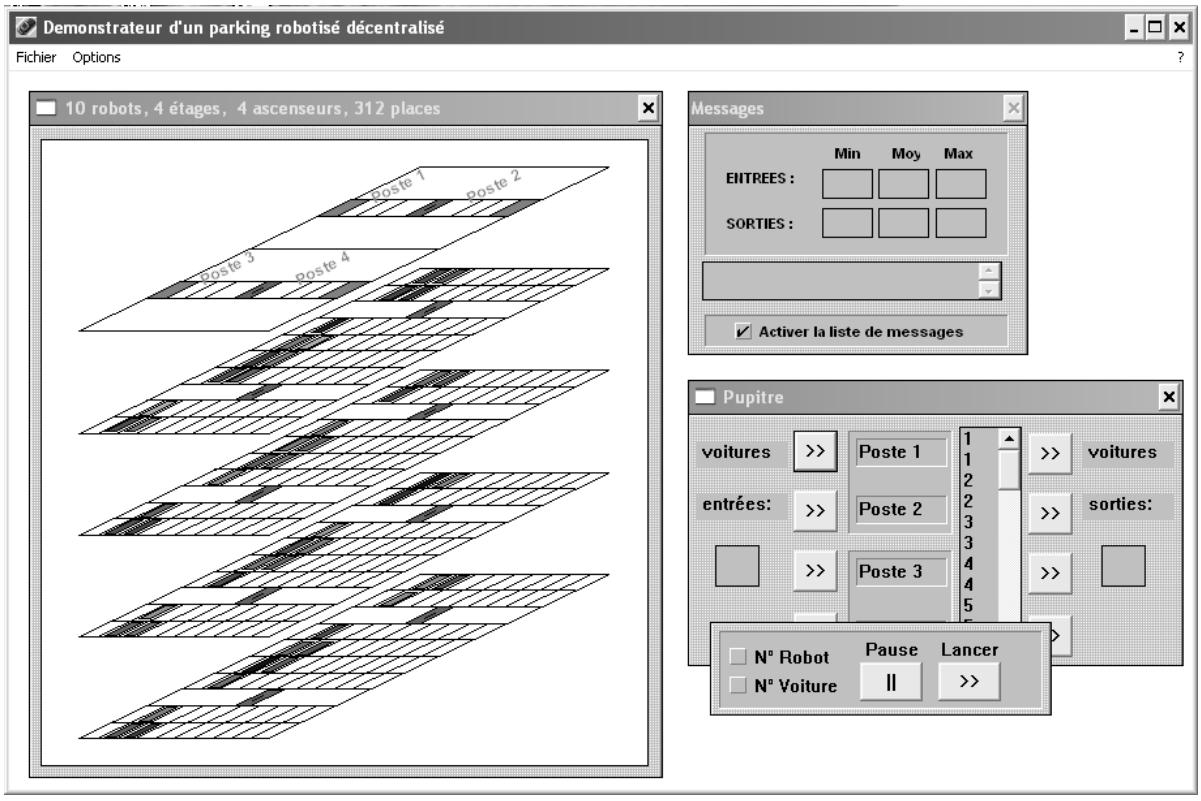

Figure 10: Display window of the simulation model

4 Conclusions

In this paper, we have seen that we can characterize a complex system of 'mechatronics' when we can confer to each one of its components the sufficient decisional capacity, for its own behavior and for the collective behavior. This approach allows, without having to implement centralized systems for scheduling and trajectory planning, to obtain the optimized trajectories, i.e. the shortest possible. This makes it possible to minimize the waiting of customers, who have, in this way, a better quality of service. This approach can be applied to various applications *(Broissin, N., 1999; Pujo et al., 1996; Broissin et al., 1996)*, that are equivalent to systems mechatronics composed of mechatronics machines. This approach represents a strong research and development potential for automated and robotized systems.

5 References

Anselmetti, B., "Algorithme de génération automatique d'une phase de tournage", *La gamme automatique en usinage*, groupe GAMA, Hermès, 1990

Bongaerts, L., Monostori, L., McFarlane, D., and Kadar, B., "Hierarchy in distributed shop floor control", *Computers In Industry, vol 43, 2000, pp 123-137.*

Broissin N., *Contribution à l'amélioration de la réactivité des systèmes de production automatisés et flexibles grâce à un pilotage basé sur une génération de tâches décentralisée*, PhD en Sciences, Université d'Aix-Marseille III, 19 juillet 1999.

Broissin, N., Pujo, P. and Bertrand, J.C., "Decentralized piloting of transport system by autonomous vehicles", *WAC 96 - World Automation Congress - ISRAM 96*, pp 125-130, Montpellier, 27 - 30 mai 1996

FIPA, "FIPA Contract Net Interaction Protocol Specification", *Foundation for Intelligent Physical Agents, December 3, 2002.*

IEEE P1516 Draft Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.

IEEE P1516.1, Draft Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification.

IEEE P1516.2, Draft Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification.

IMS, 4th IFAC WS on Intelligent Manufacturing Systems, *IMS'97, July 21-23, Seoul, Korea - Proceedings edited by Jongwon K., PERGAMON, ISBN 0-08-043025-2.*

Kanchanasevee,P., Biswas, G., Kawamura, K. and Tamura, S., "Contract-Net Based Scheduling for Holonic Manufacturing Systems", *Proceedings of SPIE ANISMI, 15-16 October 1997, pp. 108-115.*

Koestler A., *The ghost in the machine*, Editions Arkana, Londres, 1989

Pujo, P., Broissin, N. and Bertrand, J.C., "Simplification de la programmation de tâches dans les Systèmes Automatisés de Production Flexible, par l'intégration de primitives métier dans les machines de production", *5ème Congrès International de Génie Industriel, pp 53-61, Grenoble, 2 - 4 avril 1996.*

Smith R.G., "The contract net protocol: high level communication and control in a distributed problem solver", *IEEE Trans. on Computers, Vol. 29, 1980, pp 1104-1113*.